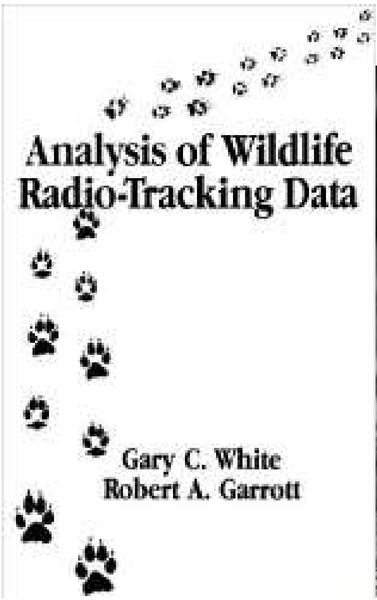
Exercise 1: The Known-Fate Model

This exercise closely follows the content of the 4th lecture and is mostly intended to show how to run program MARK to compute survival from 'known-fate' data.. We have prepared exercises for use in either MARK or RMark.



Data input

Input data consists of summarized frequencies of encounter-histories. Encounter-history is distinguished here from capture-history by having two codes for each occasion - One for capture-status, and one for recapture-status. The possibilities here are: 11=captured and dead recovery, and 10=captured and not recovered as dead.

The example we'll use for this model contains 2 groups of animals: Treatment and Control. So, each encounter-history will have 2 frequencies following the encounter-history. The following is a sample MARK input file (knownfate.inp):

```
/* known fate example (pg 344) */
10 19 21   /* 19 treatment-alive, 21 control-alive*/;
11 38 38   /* 38 treatment-dead,  38 control-dead */;
```

Explanation: The '10 19 21' means that 19 animals from group 1 (treatment) and 21 animals from group 2 (control) were captured in time 1 and were not recovered as dead (ie. they survived). The '11 38 38' means that 38 animals from group 1 and 38 animals from group 2 were captured in time 1 and recovered as dead in time 2 (ie. they did not surive).

For R users

1) Open the file "Known_Fate.r" in your preferred text editor for R (e.g., R Studio, Tinn-R, Notepad++). Note that the hash (#) symbol denotes comments in R and everything on the line after that symbol is ignored by R. The first non-comment lines in the file clear the R workspace (memory) and set the working directory(folder) to the folder where you have the exercises. You will need to modify the working directory. Execute these lines in R. (In R studio or Tinn-R, place the cursor on the first of these lines, then click the 'Run' button 3 times. In Notepad++, highlight those 3 lines, copy to clipboard, then paste into R window.)

```
rm(list=ls())                                  #   clear workspace
setwd('h:/x/workshops/uf2016/exercises')  # set working directory(folder)
library(RMark)
```

2) Assuming that you have already created a MARK input file, the first step in using RMark is to convert the input file into an RMark input "data-frame". (You can think of a data-frame as a spreadsheet in R, where you can access the spreadsheet by rows and columns, or by variable(column) name. This conversion is done with the "convert.inp" function. The only required argument is the input filename (knownfate.inp), but since we have 2 groups in our input file, we also need to specify the group names to the convert.inp function. This is done with the group.df argument. The converted data-frame is saved with the R variable name, "mdeerinp". Execute that line, then type the new variable name into the R window.

```
mdeerinp= convert.inp('knownfate.inp',group.df=data.frame(trt=c('trt','cnt')))
mdeerinp
```

3) The 2nd step is to create a processed-data variable which contains other variables needed to setup and run the MARK models. The variable, "mdeerpr" is a "list" variable, or a variable which contains other variables. The RMark function to do this is "process.data" and requires the converted input from the previous step, the type of MARK models which will be run, and the group variable name (if applicable). Execute this line, then type 'mdeerpr' to see the contents of this variable.

```
mdeerpr = process.data(mdeerinp, model='Known', groups="trt")

mdeerpr
```

4) The 3rd step is to create design-matrix data variables, needed by MARK to build models. The RMARK function is "make.design.data" and the processed-data variable created in the previous step is needed as an argument to the function. Execute this line and type 'mdeerdd' to view the contents of this variable. This is a list-type variable and it contains a data-frame variable (S) for

the estimated parameter for the model-type (Known) we specified. The columns of S are the variables we can use in building MARK models.

```
mdeerdd = make.design.data(mdeerpr)
mdeerdd
```

Modeling strategy is to develop 2 models. One represents the null hypothesis that there is no difference between the survival rates of the deer in the 2 treatments. The other model represents the alternative hypothesis that the treatment and control groups show different survival rates (control survival > treatment survival).

5) Now we're ready to build our first MARK model. With only one survival interval in the data, we don't have a lot of choices in building models. We'll start with the most simple model (survival equal among treatment and control groups). To create a model, we call the 'make.mark.model' function with the processed-data variable created in step 3 (mdeerpr), the design-data variable created in step 4 (mdeerdd), title, and list of parameters as arguments. In this case, there is only 1 parameter (S) and we specify the model using an R formula. Here, formula=~1 means the parameter is a constant value. Execute these lines and type 'mod_null' to examine what we've created. This variable is another list-variable which contains other variables to be used when we run the model.

```
Mod_null=make.mark.model(
  mdeerpr,mdeerdd,title='MuleDeerdata',
  parameters=list(
    S=list(formula=~1)
  )
)
mod_null
```

6) We can run the model by calling the 'run.mark.model' function with the model variable created in step 5 as the argument. Execute this line then type 'mod_null_out' to examine the contents. The output from MARK is stored in a text file and by typing 'mod_null_out', the text file is opened in notepad. This is the usual output you would get if you ran MARK interactively using its GUI. The output is also stored as an R list-variable. Type 'str(mod_null_out)' to display the structure of this list-variable. What is the survival rate estimate from this model? You can get it from the text output displayed in notepad, or by looking at the mod_null_out$results$real$estimate variable.

```
mod_null_out=run.mark.model(mod_null)
str(mod_null_out)
print(mod_null_out)
print(mod_null_out$results$real)
```

7) To run another model, we only need to repeat the last 2 steps: create.mark.model, and run.mark.model. With these data, the interest was in whether the treatment affected survival, so we'd like to try a model where survival is different for control and treatment groups. So, we make a new mark model, named mod_trt, where survival is different among the 2 groups (S=list(formula=~group)). Execute these lines, as well as the line to run the model (mod_trt_out=run.mark.model…). What are the survival rate estimates for the 2 groups? Are they different? Are they significantly different?

```
mod_trt = make.mark.model(mdeerpr,mdeerdd,parameters=list(
 S=list(formula=~group)
))
mod_trt_out = run.mark.model(mod_trt)
print(mod_trt_out$results$real)
```

8) We can create a table of model results by calling the 'model.table' function, with the names of the variables which contain the output of each model as an argument. Execute this line, and the next line to print the table. Assuming AIC was covered in the lecture, what does the table tell you about the effect of treatment on survival rates? Do a likelihood-ratio test between models. Does the likelihood-ratio test lead to the same conclusion as model selection?

```
#           create AIC table of model results for model comparison
tbl=model.table(model.list=c("mod_null_out","mod_trt_out"),type="Known")
print(tbl)
```

Questions:

1. Which model would you choose to best describe these data? Why?
2. Does a "z-test" comparison of survival rates between the 2 groups agree with the AIC model selection table?
    a. $Z = (S_{cnt}\text{-}S_{trt})/(var(S_{cnt})+var(S_{trt}))$
    b. Need R function, pnorm
3. Do the probability levels associated with the z-test and likelihood ratio test differ? If so, why?
    a. LR=Pr(Chi-sq, k), where Chi-sq = $L_{null}\text{-}L_{trt}$ and k=$df_{null}\text{-}df_{trt}$
    b. Need R function, pchisq